CS 584
Data Mining

Classification 4

Today

- Additional classifiers
  - Naïve Bayes classifier
  - Support Vector Machine
  - Ensemble methods
- Clustering

Recall: Bayesian Classifiers

- Approach:
  - compute the posterior probability $P(C \mid A_1, A_2, \ldots, A_n)$ for all values of C using the Bayes theorem

$$P(C \mid A_1 A_2 \ldots A_n) = \frac{P(A_1 A_2 \ldots A_n \mid C) P(C)}{P(A_1 A_2 \ldots A_n)}$$

  - Choose value of C that maximizes
    $P(C \mid A_1, A_2, \ldots, A_n)$

  - Equivalent to choosing value of C that maximizes
    $P(A_1, A_2, \ldots, A_n \mid C)\, P(C)$

- How to estimate $P(A_1, A_2, \ldots, A_n \mid C)$?

How to Estimate Probabilities from Data?

Tid	Home Owner	Marital Status	Annual Income	Defaulted
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

- Class: $P(C) = N_c/N$
  - i.e., P(No) = 7/10,
    P(Yes) = 3/10
- For discrete attributes:

$$P(A_i \mid C_k) = |A_{ik}| / N_c$$

  - where $|A_{ik}|$ is number of instances having attribute A_i and belongs to class C_k
  - Examples:

    P(MaritalStatus=Married|No) = 4/7
    P(HomeOwner=Yes|Yes)=0

- To simplify the task, **naïve Bayesian classifiers** assume attributes have independent distributions, and thereby estimate

$$P(A|C) = P(A_1|C) * P(A_2|C) * \ldots * P(\mathrm{A_n}|C)$$

The probability of class *C* generating instance *A*, equals….

The probability of class *C* generating the observed value for feature 1, multiplied by..

The probability of class *C* generating the observed value for feature 2, multiplied by..

How to Estimate Probabilities from Data?

- For continuous attributes:
  - Discretize the range into bins
    - one ordinal attribute per bin
    - violates independence assumption
  - Two-way split: $(A < v)$ or $(A > v)$
    - choose only one of the two splits as new attribute
  - Probability density estimation:
    - Assume attribute follows a normal distribution
    - Use data to estimate parameters of distribution (e.g., mean and standard deviation)
    - Once probability distribution is known, can use it to estimate the conditional probability $P(A_i|C)$

k

How to Estimate Probabilities from Data?

Tid	Home Owner	Marital Status	Annual Income	Defaulted
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

- Normal distribution:

$$P(A_i \mid C) = \frac{1}{\sqrt{2\pi\sigma_{ij}^2}} e^{-\frac{(A_i-\mu_{ij})^2}{2\sigma_{ij}^2}}$$

  - One for each (A_i, C) pair

- For (Income, Class=No):
  - If Class=No
    - sample mean = 110
    - sample variance = 2975

$$P(Income = 120 \mid No) = \frac{1}{\sqrt{2\pi}(54.54)} e^{-\frac{(120-110)^2}{2(2975)}} = 0.0072$$

More Example

Given a Test Record:

$$X = (\text{HomeOwner} = \text{No}, \text{Married}, \text{Income} = 120\text{K})$$

naive Bayes Classifier:

P(HomeOwner=Yes|No) = 3/7
P(HomeOwner = No|No) = 4/7
P(HomeOwner = Yes|Yes) = 0
P(HomeOwner = No|Yes) = 1

P(Marital Status=Single|No) = 2/7
P(Marital Status=Divorced|No)=1/7
P(Marital Status=Married|No) = 4/7
P(Marital Status=Single|Yes) = 2/3
P(Marital Status=Divorced|Yes)= 1/3
P(Marital Status=Married|Yes) = 0

For taxable income:
If class=No: sample mean=110
sample variance=2975
If class=Yes: sample mean=90
sample variance=25

- P(X|Class=No) = P(HomeOwner=No|Class=No)
  × P(Married| Class=No)
  × P(Income=120K| Class=No)
  = 4/7 × 4/7 × 0.0072 = 0.0024

- P(X|Class=Yes) = P(HomeOwner=No| Class=Yes)
  × P(Married| Class=Yes)
  × P(Income=120K| Class=Yes)
  = 1 × 0 × 1.2 × 10^{-9} = 0

Since P(X|No)P(No) > P(X|Yes)P(Yes)

Therefore P(No|X) > P(Yes|X)

=> Class = No

Naïve Bayes Classifier

- If one of the conditional probability is zero, then the entire expression becomes zero
- Probability estimation:

$$\text{Original}: P(A_i \mid C) = \frac{N_{ic}}{N_c}$$

$$\text{Laplace}: P(A_i \mid C) = \frac{N_{ic} + 1}{N_c + c}$$

$$\text{m - estimate}: P(A_i \mid C) = \frac{N_{ic} + mp}{N_c + m}$$

N_c = # of samples from class c

N_{ic} = # of sample from class c that takes on value i

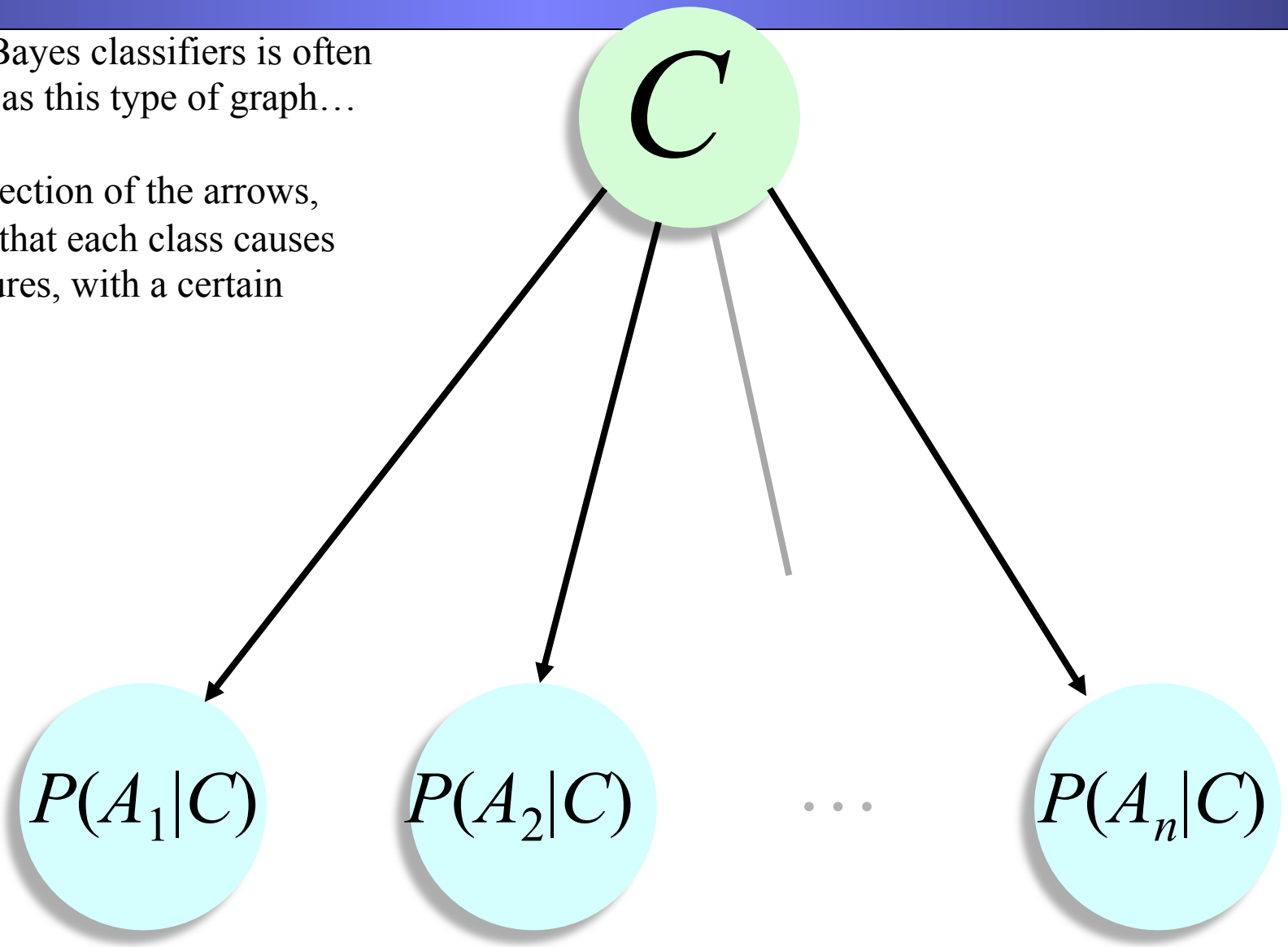
c: number of classes

p: prior probability of P

m: parameter, measured in # of samples, it says how confident we're of our prior estimate of p.

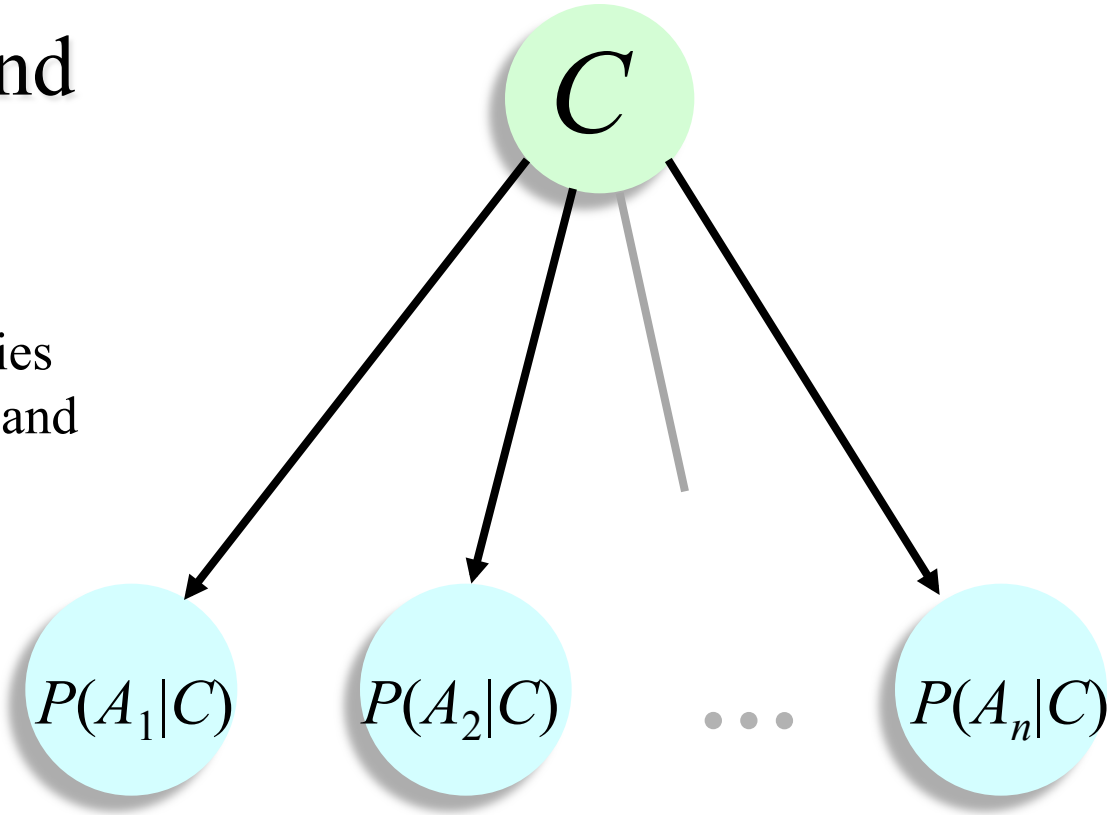
The Naive Bayes classifiers is often represented as this type of graph…

Note the direction of the arrows, which state that each class causes certain features, with a certain probability



Naïve Bayes is fast and space efficient

We can compute all the probabilities with a single scan of the database and store them in a (small) table…



Sex	Over190$_{cm}$	
Male	Yes	0.15
	No	0.85
Female	Yes	0.01
	No	0.99

Sex	Long Hair	
Male	Yes	0.05
	No	0.95
Female	Yes	0.70
	No	0.30

Sex	
Male	
Female	

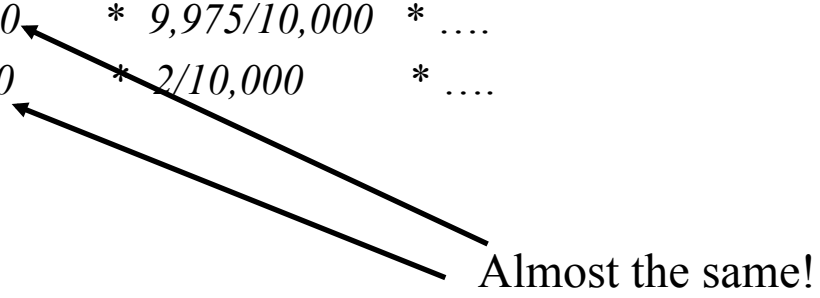
Naïve Bayes is NOT sensitive to irrelevant features...

Suppose we are trying to classify a person's sex based on several features, including eye color. (Of course, eye color is completely irrelevant to a persons gender)

$$P(\text{Jessica} \mid C) = P(\text{eye} = \text{brown} \mid C) * P(\text{wears\_dress} = \text{yes} \mid C) * \ldots.$$

$p(\text{Jessica} \mid \text{Female}) = 9{,}000/10{,}000 * 9{,}975/10{,}000 * \ldots.$

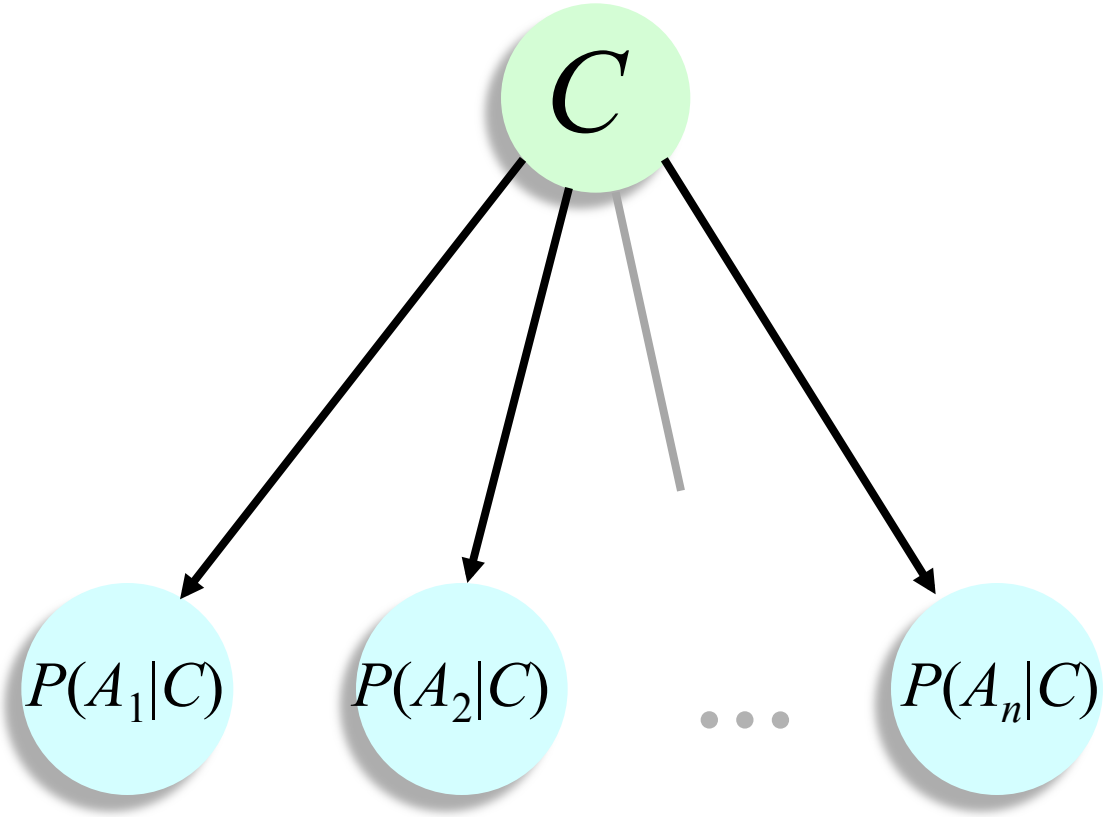
$p(\text{Jessica} \mid \text{Male}) = 9{,}001/10{,}000 * 2/10{,}000 * \ldots.$



However, this assumes that we have good enough estimates of the probabilities, so the more data the better.

Problem!

Naïve Bayes assumes independence of features…

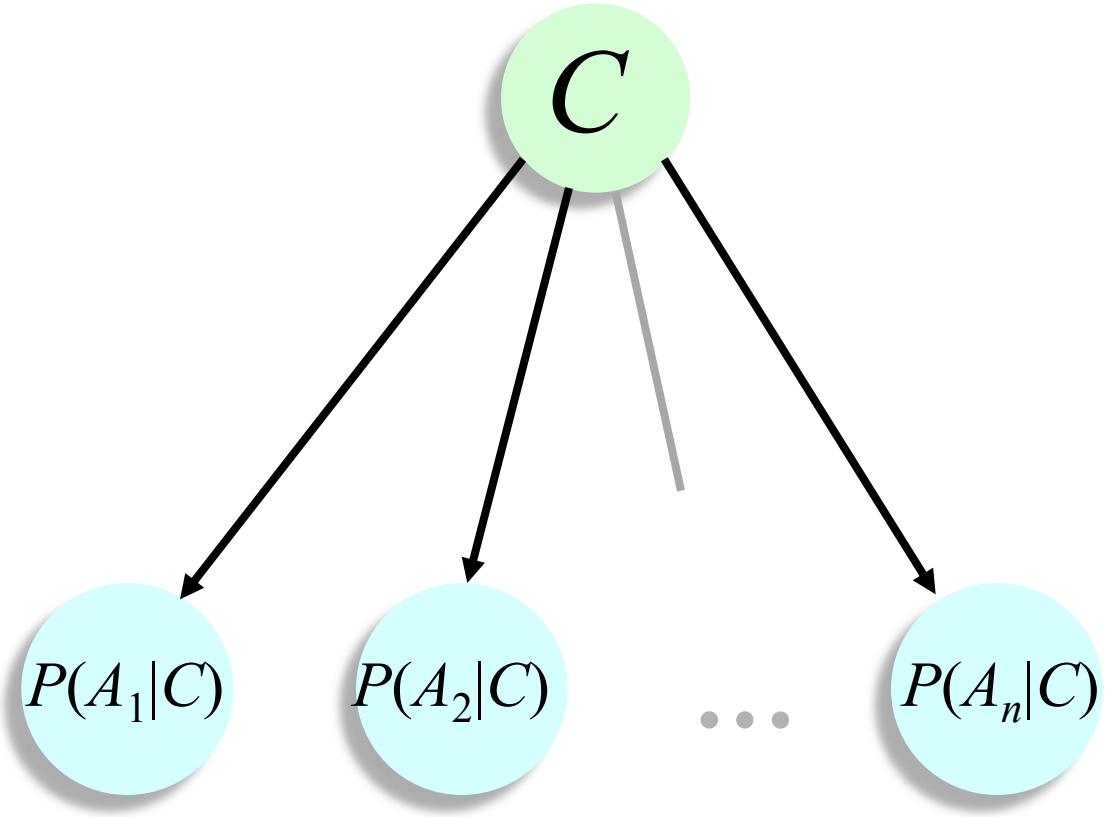


Sex	Over 6 foot	
Male	Yes	0.15
	No	0.85
Female	Yes	0.01
	No	0.99

Sex	Over 200 pounds	
Male	Yes	0.11
	No	0.80
Female	Yes	0.05
	No	0.95

Solution

Consider the relationships between attributes…

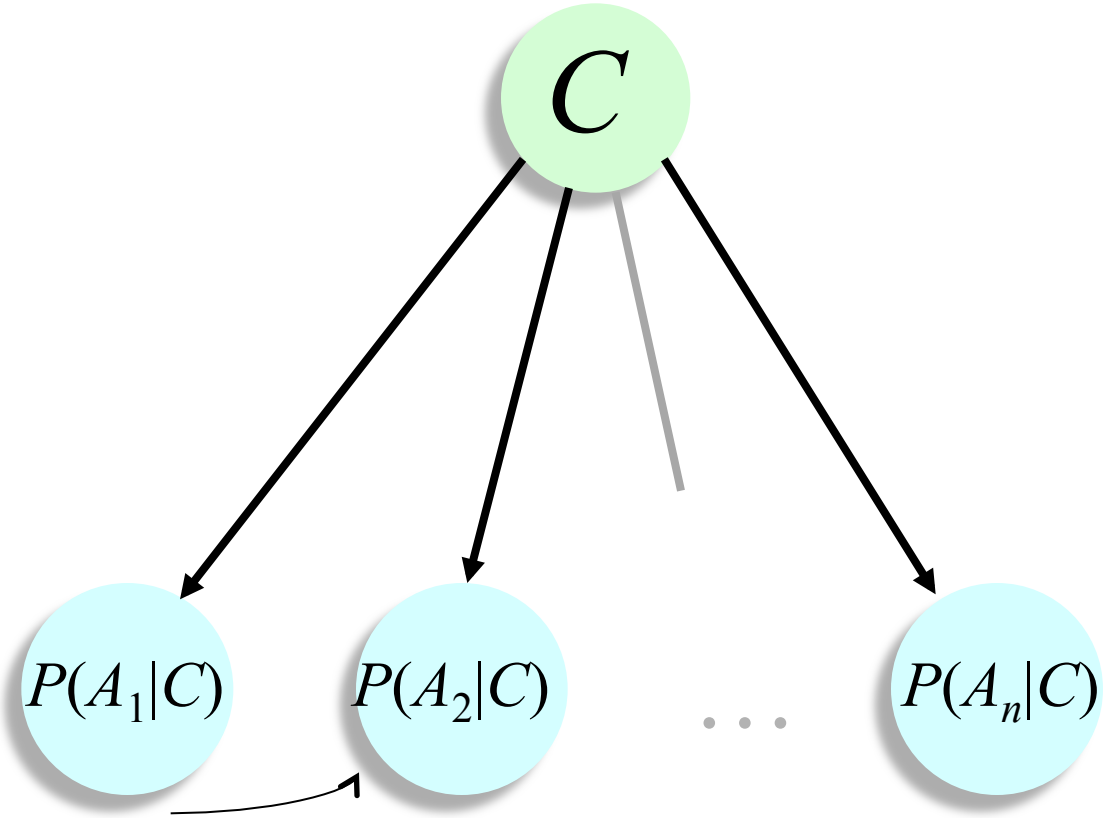


Sex	**Over 6 foot**	
Male	Yes	0.15
	No	0.85
Female	Yes	0.01
	No	0.99

Sex	**Over 200 pounds**	
Male	Yes and **Over 6 foot**	0.11
	No and **Over 6 foot**	0.59
	Yes and NOT **Over 6 foot**	0.05
	No and NOT **Over 6 foot**	0.35
Female	Yes and **Over 6 foot**	0.01

Solution

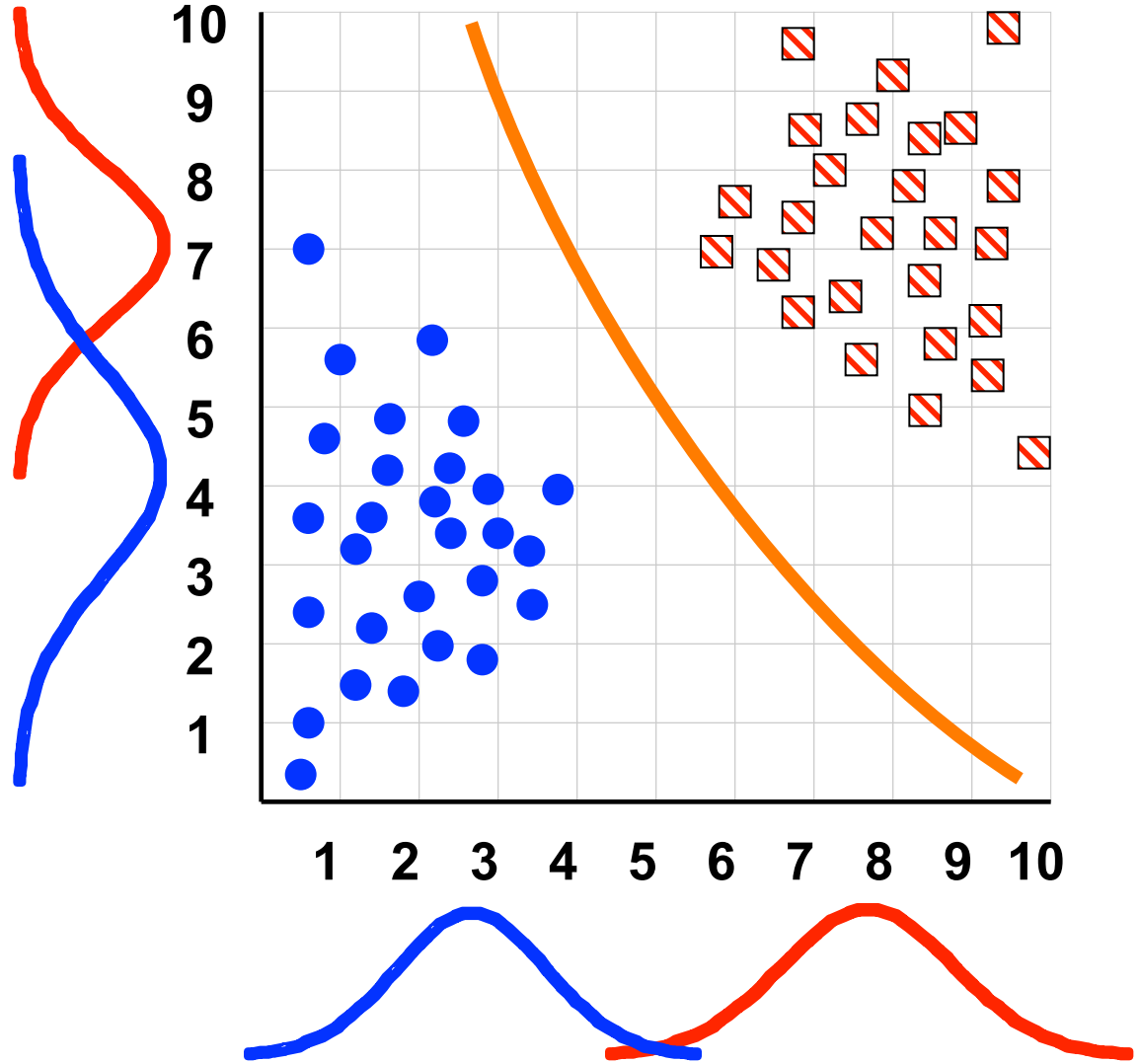
Consider the relationships between attributes…



But how do we find the set of connecting arcs??

Use Bayesian Belief Networks

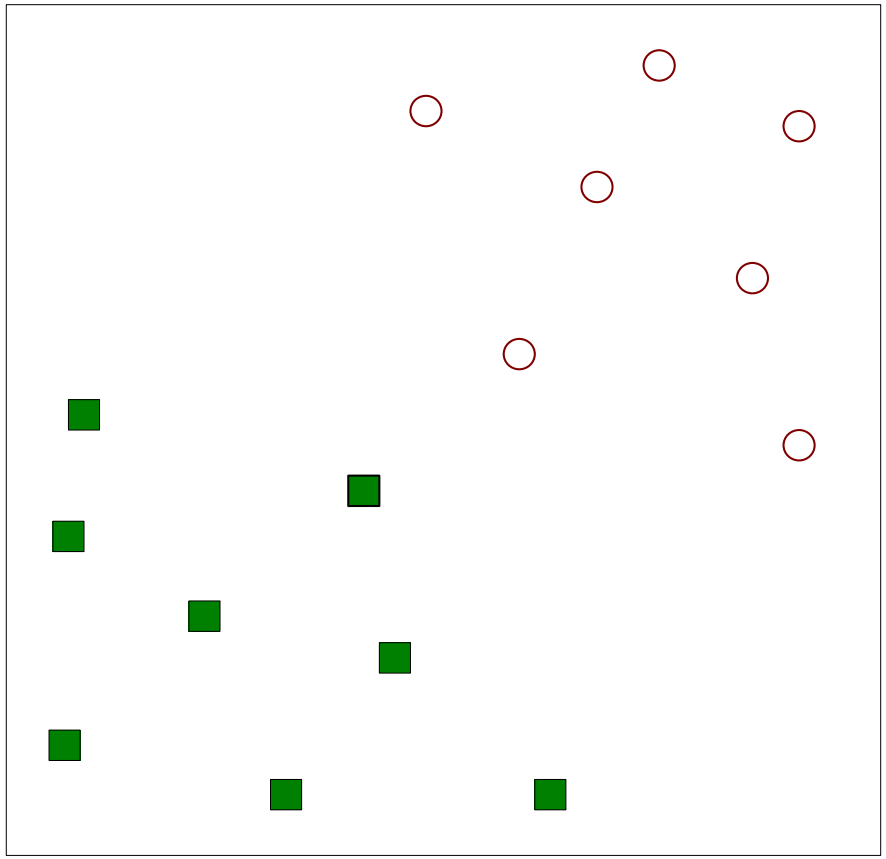
The Naïve Bayesian Classifier has a quadratic decision boundary



Advantages/Disadvantages of Naïve Bayes

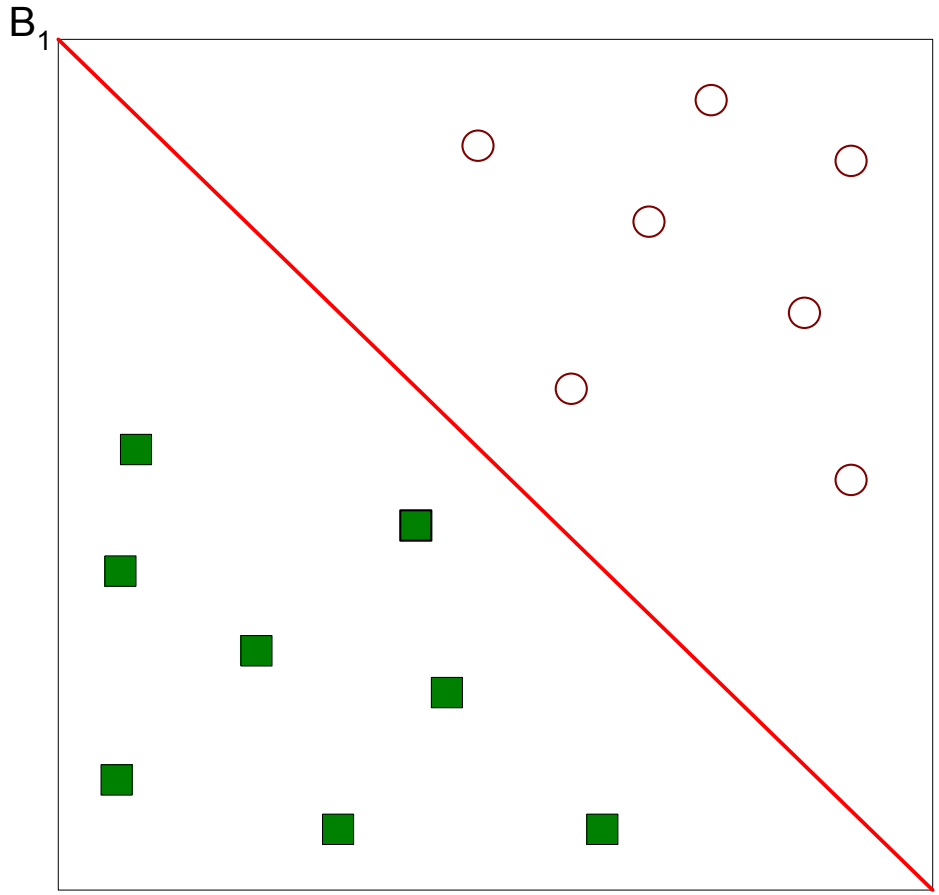
- Advantages:
  - Fast to train (single scan). Fast to classify
  - Not sensitive to irrelevant features
  - Robust to isolated noise points
  - Handles real and discrete data
  - Handles streaming data well
  - Handle missing values by ignoring the instance during probability estimate calculations
- Disadvantages:
  - Independence assumption may not hold for some attributes
    - Use other techniques such as Bayesian Belief Networks

Support Vector Machines



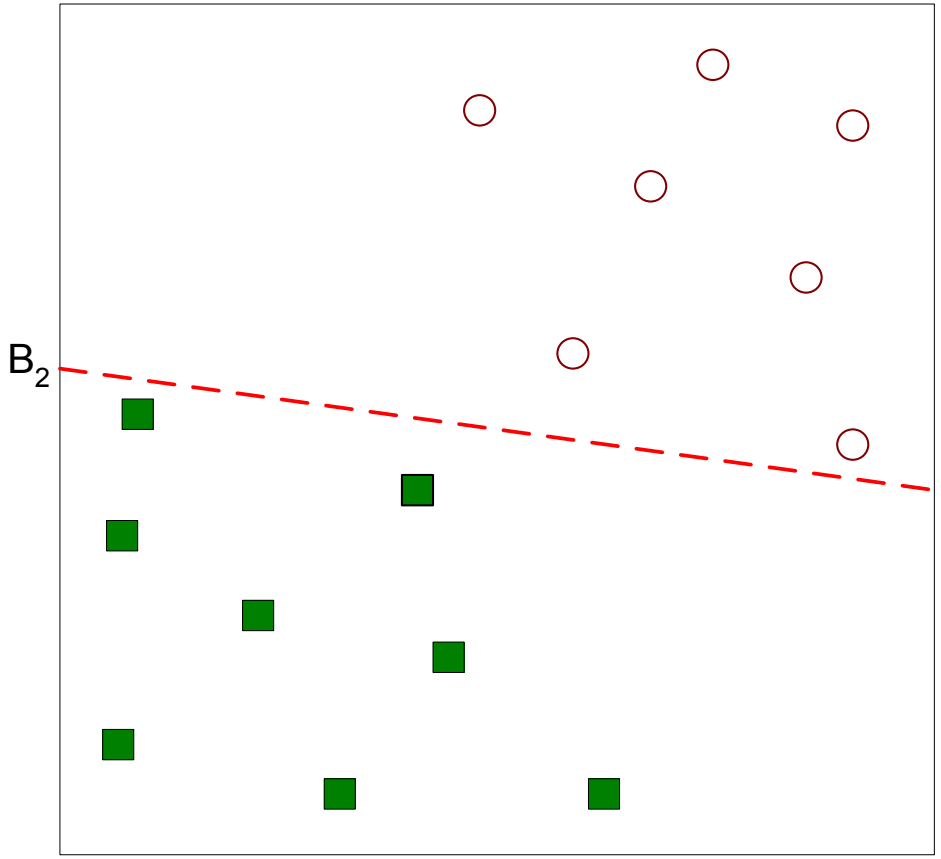
- Find a linear hyperplane (decision boundary) that will separate the data

Support Vector Machines



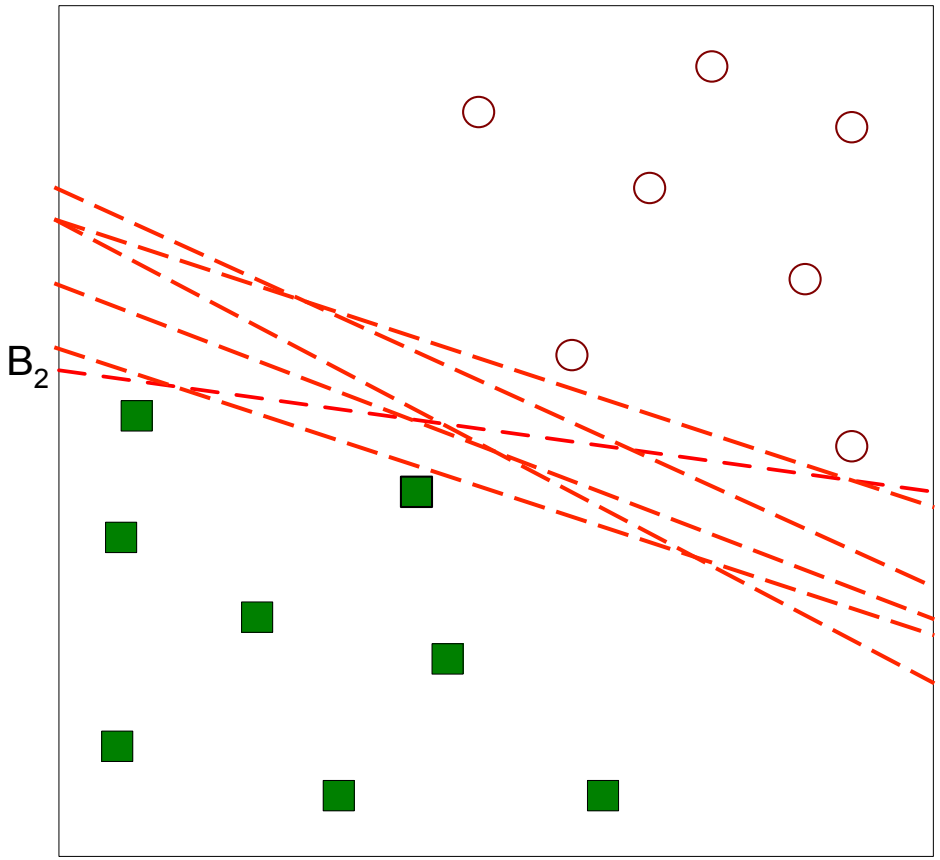
- One Possible Solution

Support Vector Machines



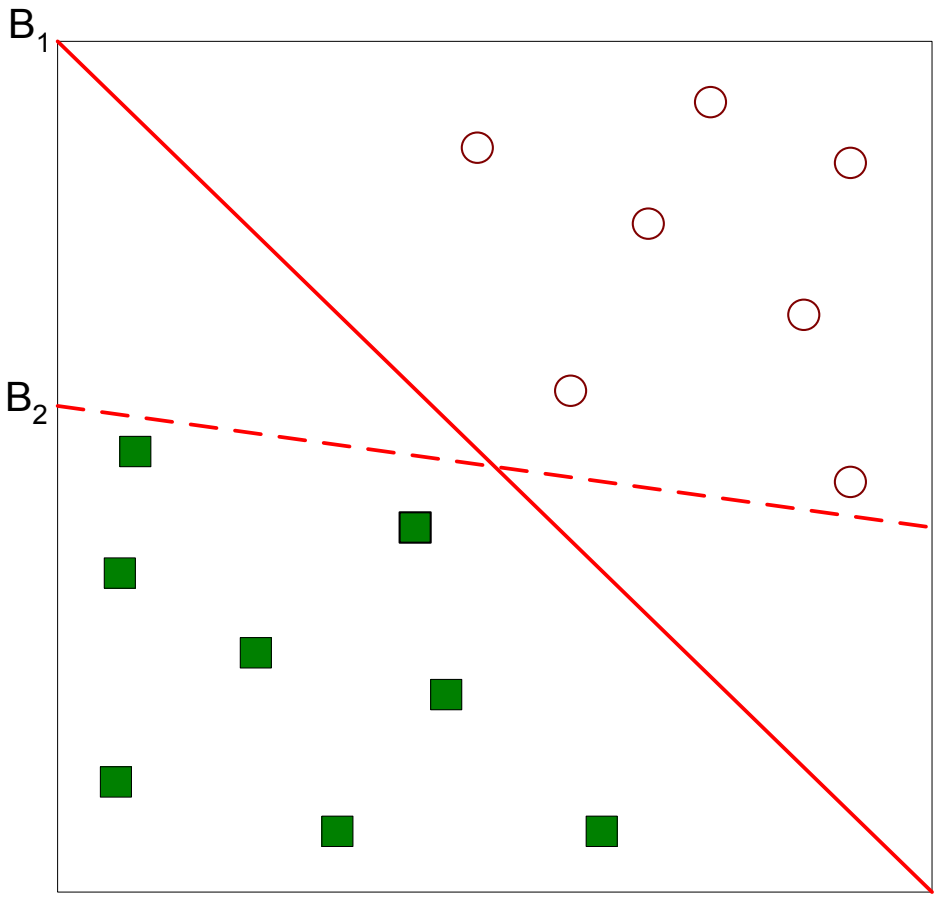
- Another possible solution

Support Vector Machines



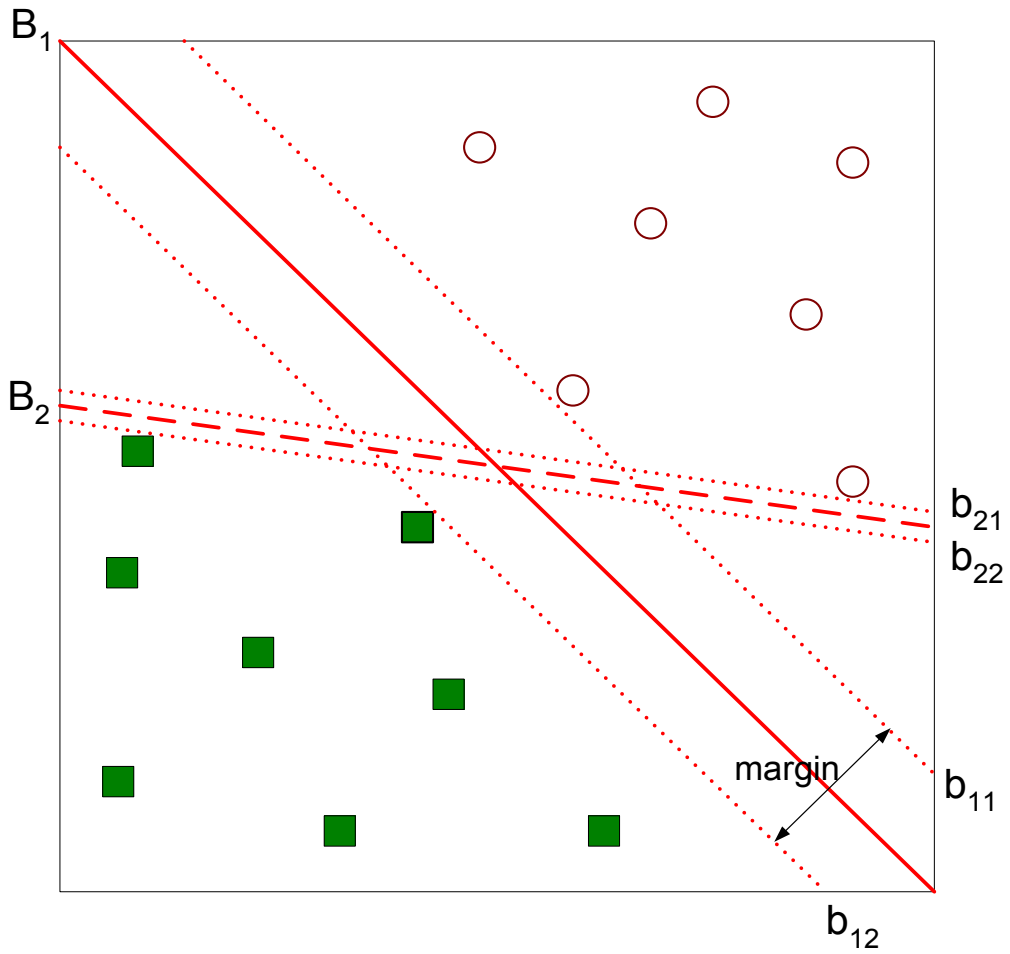
- Other possible solutions

Support Vector Machines



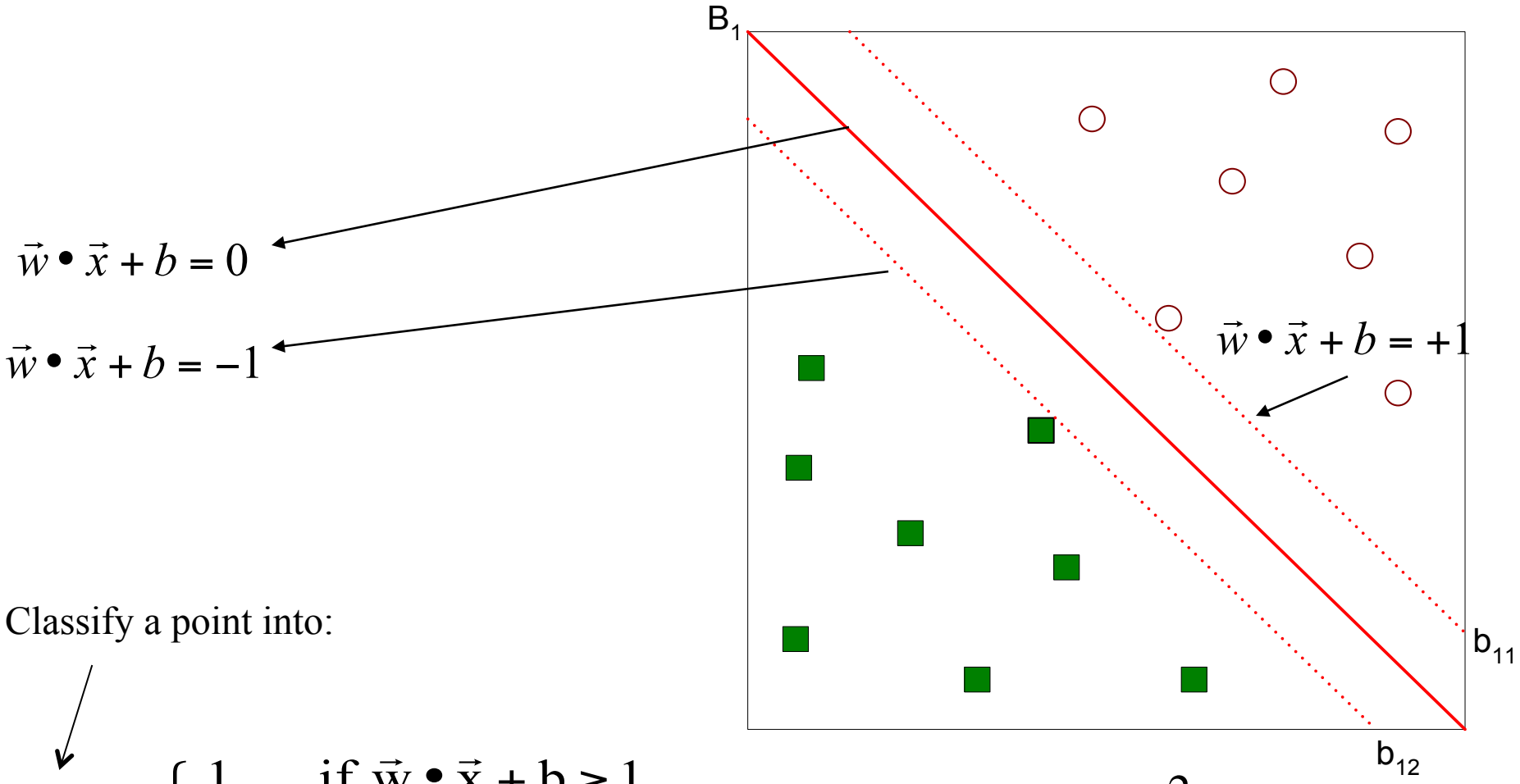
- Which one is better? B1 or B2?
- How do you define better?

Support Vector Machines



- Find hyperplane that maximizes the margin => B1 is better than B2

Support Vector Machines



Classify a point into:

$$f(\vec{x}) = \begin{cases} 1 & \text{if } \vec{w} \bullet \vec{x} + b \geq 1 \\ -1 & \text{if } \vec{w} \bullet \vec{x} + b \leq -1 \end{cases}$$

$$\text{Margin} = \frac{2}{\| \vec{w} \|}$$

Support Vector Machines

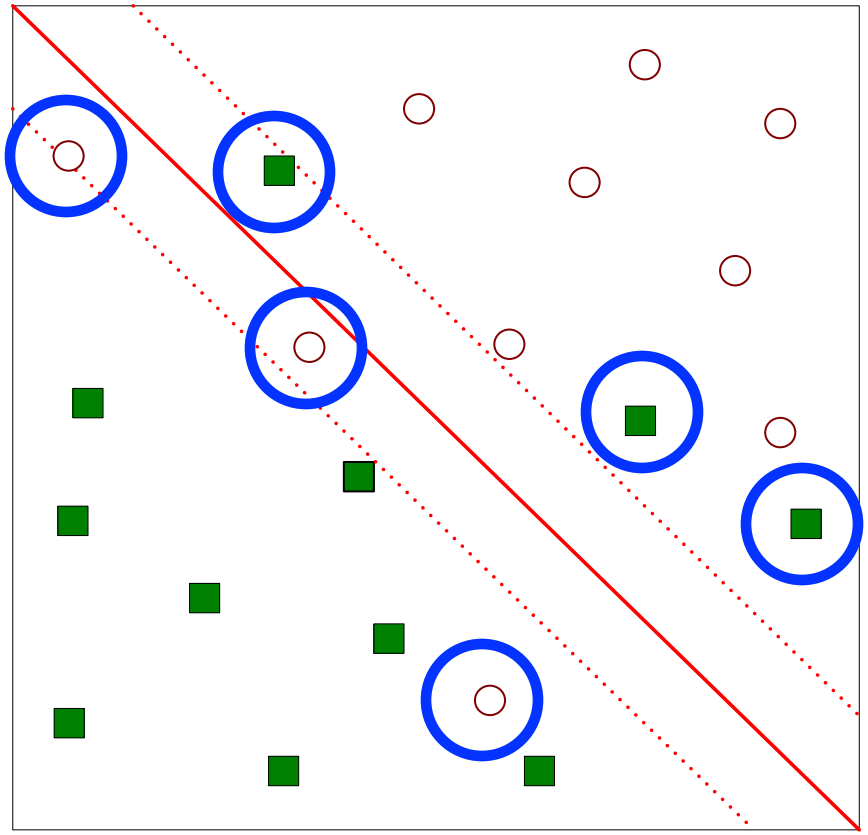
- We want to maximize: $$\text{Margin} = \frac{2}{\| \vec{w} \|}$$
  - Which is equivalent to minimizing: $$L(w) = \frac{\| \vec{w} \|^2}{2}$$
  - But subjected to the following constraints:

$$f(\vec{x}_i) = \begin{cases} 1 & \text{if } \vec{w} \bullet \vec{x}_i + b \geq 1 \\ -1 & \text{if } \vec{w} \bullet \vec{x}_i + b \leq -1 \end{cases}$$

    - This is a constrained optimization problem
      - Numerical approaches to solve it (e.g., quadratic programming)

Support Vector Machines

- What if the problem is not linearly separable?



Support Vector Machines

- What if the problem is not linearly separable?
  - Introduce slack variables
    - Need to minimize:

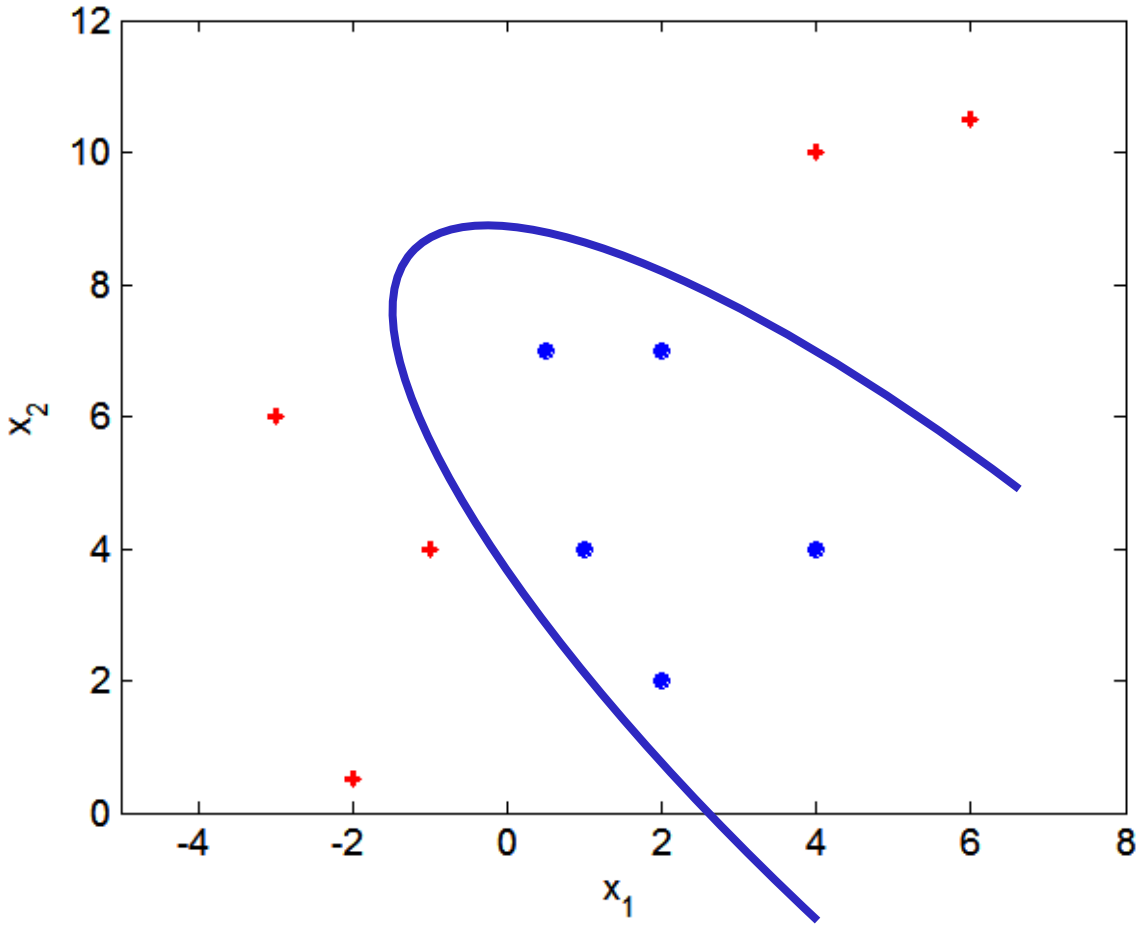
$$L(w) = \frac{\| \vec{w} \|^2}{2} + C\left(\sum_{i=1}^{N} \xi_i^k \right)$$

    - Subject to:

$$f(\vec{x}_i) = \begin{cases} 1 & \text{if } \vec{w} \bullet \vec{x}_i + b \geq 1 - \xi_i \\ -1 & \text{if } \vec{w} \bullet \vec{x}_i + b \leq -1 + \xi_i \end{cases}$$

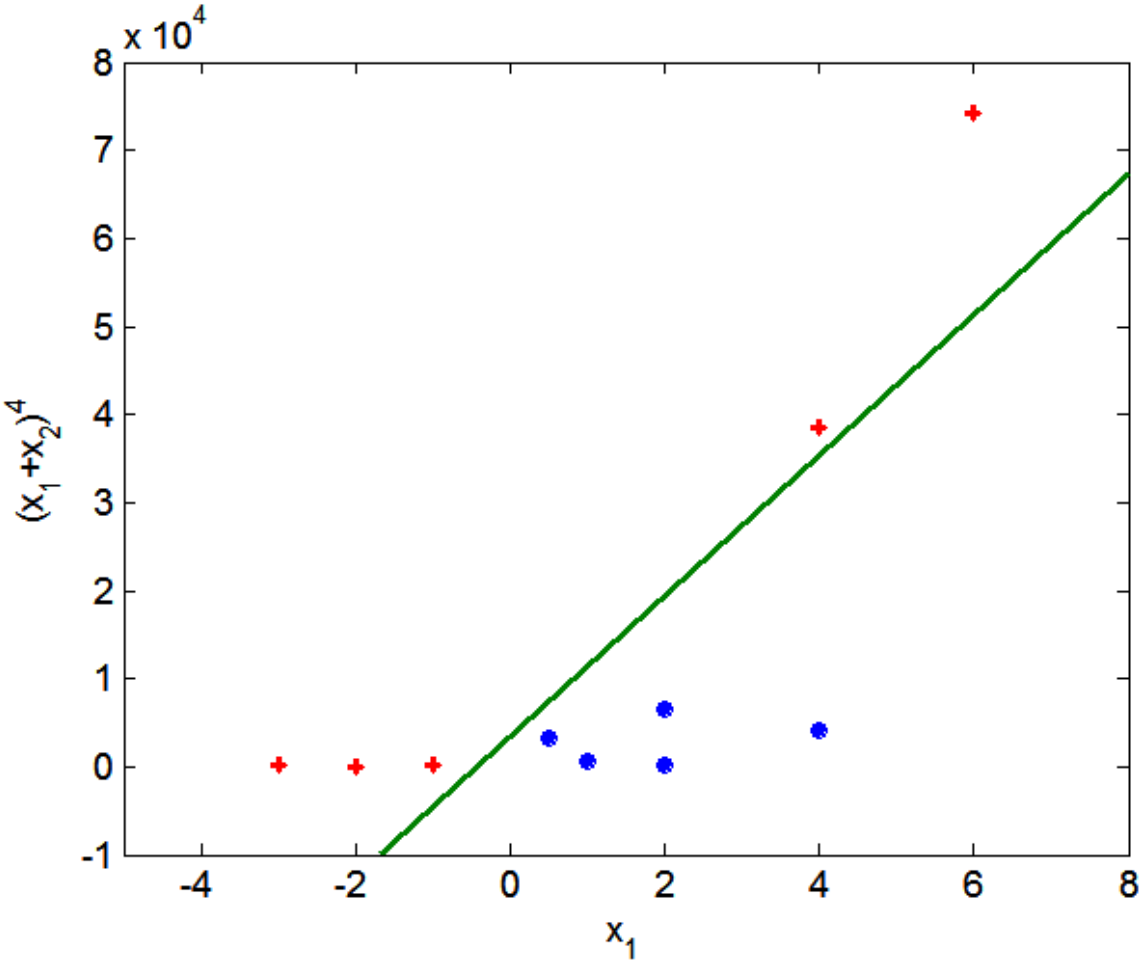
Nonlinear Support Vector Machines

- What if decision boundary is not linear?



Nonlinear Support Vector Machines

- Transform data into higher dimensional space



Why SVMs?

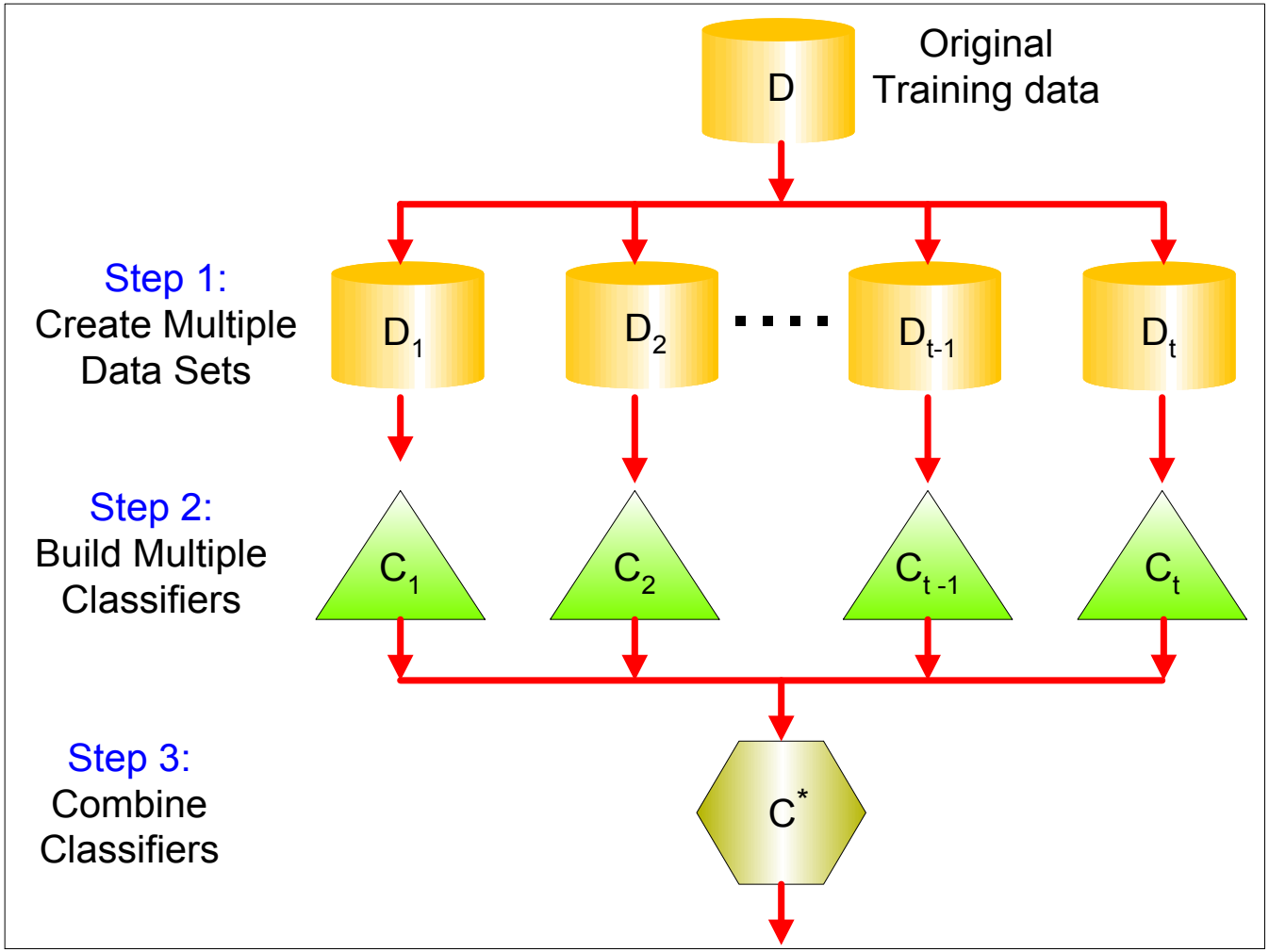
- Convex Convex Convex
  - No trapping in local minima
- SVMs work for categorical and continuous data.
- Can control the model complexity by providing the control on cost function, margin parameters to use.
- Kernel Trick (Not discussed) extends it to non-linear spaces.

Ensemble Methods

- Construct a set of classifiers from the training data

- Predict class label of previously unseen records by aggregating predictions made by multiple classifiers

General Idea



Why does it work?

- Suppose there are 25 base classifiers
  - Each classifier has error rate, ε = 0.35
  - Assume classifiers are independent
  - Probability that the ensemble classifier makes a wrong prediction:

$$\sum_{i=13}^{25} \binom{25}{i} \varepsilon^i (1-\varepsilon)^{25-i} = 0.06$$

  - Two important conditions for an ensemble classifier to perform better than a single classifier:
    - The base classifiers should be independent of each other
    - The base classifiers should do better than random guessing.

Examples of Ensemble Methods

- How to generate an ensemble of classifiers?
  - Bagging
  - Boosting

Bagging

- Bootstrap Aggregation
  - Create classifiers by drawing samples of size equal to the original dataset. (Appx 63% of data will be chosen)
  - Learn classifier using these samples.
  - Vote on them.
- Why does this help ?
  - If there is a high variance i.e. classifier is unstable, bagging will help to reduce errors due to fluctuations in the training data.
  - If the classifier is stable i.e. error of the ensemble is primarily by bias in the base classifier -> may degrade the performance.

Boosting

- An iterative procedure to adaptively change distribution of training data by focusing more on previously misclassified records
  - Initially, all N records are assigned equal weights
  - Unlike bagging, weights may change at the end of boosting round

Adaboost (Freund et. al. 1997)

- Given a set of n class-labeled tuples $(x_1,y_1) \ldots (x_n,y_n)$ i.e T
- Initially all weights of tuples are set to same (1/n)
- Generate k classifiers in k rounds. At the i-th round
  - Tuples from T are sampled from T to form training set T_i
  - Each tuple's chance of selection depends on its weight.
  - Learn a model M_i from T_i
  - Compute error rate using T_i
  - If tuple is misclassified its weight is increased.
- During prediction use the error of the classifier as a weight (vote) on each of the models

Why boosting/bagging?

- Improves the variance of unstable classifiers.
  - Unstable Classifiers
    - Neural nets, decision trees
  - Stable Classifiers
    - K-NN
- May lead to results that are not explanatory.